Sistemi Operativi
2 aprile 2007
Compitino 1/A

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Si descrivano i thread nel sistema operativo Solaris.

   Risposta: Solaris fino alla versione precedente alla 9 prevede un modello di thread a due livelli in cui più thread a livello utente sono messi in corrispondenza con un numero minore o uguale di thread a livello kernel oppure è possibile vincolare un thread a livello utente ad un solo thread a livello kernel. In questo modo un programmatore può creare a livello utente diversi thread ed il sistema operativo, a seconda del numero di CPU disponibili, può mappare questi ultimi su uno o più thread a livello kernel in modo da garantire una reale concorrenza. Quindi vengono combinati i vantaggi derivanti dai thread a livello utente e da quelli a livello kernel. I primi sono convenienti per processi CPU-bound, ovvero, per processi che non necessitino di un'attività di I/O intensiva (che bloccherebbe il processo con tutti i suoi thread) in cui sia possibile scomporre e parallelizzare il lavoro. I secondi invece sono convenienti per processi I/O-bound in cui è prevalente l'attività di I/O.

   A partire dalla versione 9 Solaris ha adottato il modello uno a uno.

2. (a) Che cosa si intende per processi CPU-bound e processi I/O-bound? Quali effetti si possono avere se viene data la precedenza a processi CPU-bound?

   (b) Si diano esempi di algoritmi di scheduling della CPU che privilegiano i processi I/O-bound.

   Risposta:

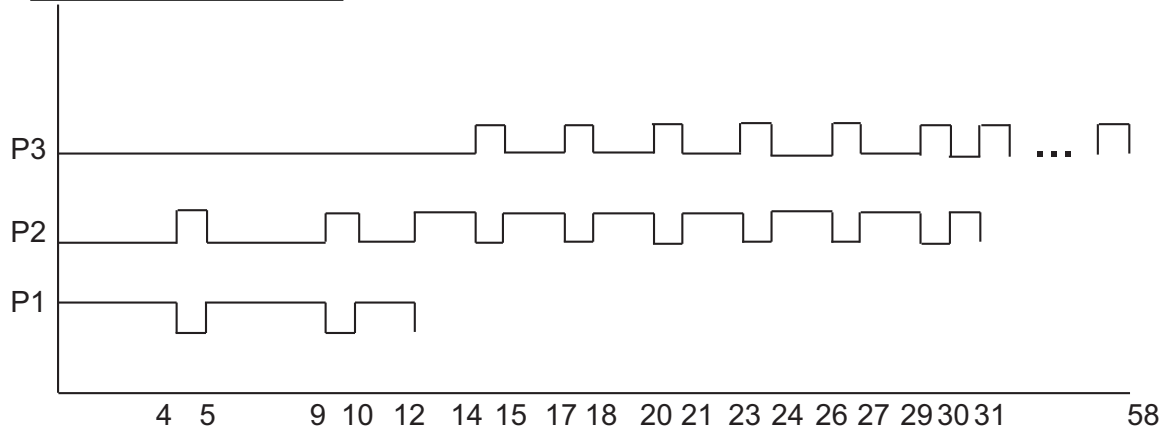
   (a) Un processo CPU-bound alterna lunghi periodi di intensiva computazione a pochi (possibilmente lunghi) cicli di I/O. Al contrario, un processo I/O-bound alterna lunghi periodi di I/O a brevi periodi di calcolo. Un possibile effetto dovuto all'accordare la precedenza ai processi CPU-bound è quello di sfavorire i processi interattivi (I/O-bound) e quindi di far percepire agli utenti del sistema un cattivo servizio dovuto a dei tempi di risposta molto alti.

   (b) Esempi di algoritmi di scheduling che privilegiano i processi I/O-bound sono SJF (Shortest Job First) e la sua variante preemptive SRTF (Shortest Remaining Time First), dato che assegnano la CPU al processo con il prossimo periodo di calcolo minimo. Un ulteriore esempio può essere un algoritmo con code multiple e prelazione fra code in base alla priorità in cui viene assegnata priorità massima a quella per processi I/O-bound.

3. I processi P_1, P_2, P_3, di priorità decrescente e con richieste di CPU totali di 10, 15, 20 s, sono in coda ready nell'ordine indicato. L'algoritmo di scheduling della CPU è con prelazione (e i processi prelazionati vengono messi in fondo alla coda ready). Ogni processo P_i ha un comportamento ciclico: in ogni ciclo consuma c_i secondi di CPU ed esegue operazioni di I/O per o_i secondi, dove $c_i = 2^{3-i}$ e $o_i = 0,5^{3-i}$. L'algoritmo di scheduling viene eseguito ad intervalli di 1 secondo. Si calcolino i tempi di attesa per i processi, nel caso l'algoritmo di scheduling sia RR con quanto pari a 2s e con prelazione in base alla priorità.

   Risposta: I tempi di attesa Δ_i dei processi P_i sono $\Delta_1 = 2s$, $\Delta_2 = 16s$ e $\Delta_3 = 38s$. Infatti abbiamo la seguente situazione:

	CPU$_t ot$	c_i	o_i
P_1	10	4	0,25
P_2	15	2	0,5
P_3	20	1	1



Sistemi Operativi
2 aprile 2007
Compitino 1/A

4. Si consideri la seguente situazione, dove P_0, P_1, P_2 sono tre processi in esecuzione, C è la matrice delle risorse correntemente allocate, Max è la matrice del numero massimo di risorse assegnabili ad ogni processo e A è il vettore delle risorse disponibili:

	C			Max		
	A	B	C	A	B	C
P_0	2	0	1	4	4	5
P_1	0	3	2	2	4	2
P_2	1	4	1	4	6	4

Available (A)

A	B	C
2	1	2

   (a) Calcolare la matrice R delle richieste.

   (b) Il sistema è in uno stato sicuro?

   Risposta:

   (a) La matrice R delle richieste è data dalla differenza $Max - C$:

R		
A	B	C
2	4	4
2	1	0
3	2	3

   (b) Sì il sistema è in uno stato sicuro in quanto esiste la sequenza sicura $\langle P_1, P_0, P_2 \rangle$. Infatti, dapprima si esegue P_1 in quanto $R_1 \leq A$ ed A diventa quindi $(2, 4, 4)$. A questo punto $R_0 \leq A$ e quindi si esegue P_0 generando il nuovo valore di A: $(4, 4, 5)$. Infine viene eseguito P_2 ($R_2 \leq A$) generando il valore finale di $A = (5, 8, 6)$.

5. Si spieghi brevemente il funzionamento dell'algoritmo del fornaio. A cosa serve?

   Risposta: L'algoritmo del Fornaio permette di risolvere il problema della sezione critica con n processi nel modo seguente:

   - si utilizza uno schema di enumerazione che generi dei numeri in ordine crescente (eventualmente anche con ripetizioni);
   - ogni processo che desidera entrare nella sezione critica riceve un numero;
   - chi possiede il numero più basso entra nella sezione critica;
   - eventuali conflitti vengono risolti da un ordinamento statico dei processi: se P_i e P_j ricevono lo stesso numero e $i < j$, allora P_i ottiene l'accesso alla sezione critica prima di P_j (viceversa se $j < i$, allora P_j accede per primo alla sezione critica).

6. In generale per prevenire i deadlock (deadlock prevention) cosa bisogna assicurare? Si faccia un esempio di una tecnica di prevenzione dei deadlock.

   Risposta: Siccome le condizioni di Coffman sono necessarie per il verificarsi di un deadlock, è sufficiente che (almeno) una di esse sia falsa perché sia automaticamente impossibile che si verifichi un deadlock. Ad esempio si può falsificare la condizione "Hold and Wait" obbligando ogni processo a dichiarare in anticipo il numero ed il tipo di risorse necessarie a completare il suo compito. In questo modo, se le risorse sono tutte disponibili, vengono assegnate al processo che può essere eseguito fino alla sua terminazione senza rischio di stallo. Nel caso in cui invece una delle risorse necessarie non sia disponibile, il processo viene posto in stato di attesa.

7. Che cos'è un mutex? Di che istruzione macchina è necessario disporre per implementare un mutex in spazio utente?

   Risposta: Un mutex è sostanzialmente una versione semplificata del semaforo quando non serve la capacità di "contare" il numero di wakeup. Quindi un mutex è una variabile che può assumere solo due valori: bloccato o sbloccato. In questo caso infatti le operazioni down e up vengono chiamate *mutex_lock* e *mutex_unlock*. Per implementare un mutex in spazio utente è necessario disporre di una speciale istruzione TSL (Test-and-Set-Lock), ovvero, un'istruzione che controlla e modifica il contenuto di una parola atomicamente. Lo pseudo codice relativo è il seguente:

```
function Test-and-Set (var target: boolean): boolean;
  begin
    Test-and-Set := target;
    target := true;
  end;
```

Questi due passi devono essere implementati in modo atomico in assembler (ovvero, il bus della memoria deve essere bloccato mentre vengono eseguite le due operazioni in modo da impedire l'accesso ad altri processi).

Sistemi Operativi
2 aprile 2007
Compitino 2/A

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Si illustri la differenza fra binding degli indirizzi a:

   1. tempo di compilazione (compile time),
   2. tempo di caricamento (load time),
   3. tempo di esecuzione (execution time).

   Quale delle tre modalità precedenti necessità di un supporto hardware?

   Risposta: L'associazione di istruzioni e dati a indirizzi di memoria può avvenire in vari momenti:

   - compile time: l'associazione avviene a tempo di compilazione e quindi si produce del codice assoluto (la locazione di esecuzione è nota a priori); nel caso in cui si voglia cambiare locazione di esecuzione, bisogna ricorrere ad una ricompilazione del codice;
   - load time: l'associazione avviene a tempo di caricamento; siccome la locazione di esecuzione non è nota a priori, il compilatore produce del codice rilocabile la cui posizione in memoria viene decisa al momento del caricamento e non può essere cambiata durante l'esecuzione del programma;
   - execution time: siccome l'associazione è stabilita a tempo di esecuzione, il programma può essere spostato da una zona all'altra della memoria mentre viene eseguito; questa modalità necessita di un supporto hardware (ad esempio registri base e limite).

2. Come è strutturata una tabella delle pagine invertita? Come si effettua la traduzione da indirizzo virtuale ad indirizzo fisico? Quando ha senso ricorrere a questa soluzione?

   Risposta: L'idea alla base di questa tecnica è di mantenere una tabella con una entry per ogni frame fisico della memoria, non per ogni pagina virtuale. Le informazioni contenute in ogni entry sono il numero della pagina (virtuale) memorizzata in quel frame ed informazioni riguardanti il processo che possiede la pagina. In questo modo diminuisce la memoria necessaria per memorizzare le page table, a discapito del tempo di accesso alla tabella stessa. Quest'ultimo ovviamente aumenta dato che, per tradurre un indirizzo virtuale nel corrispondente indirizzo fisico, è necessario scandire tutta la tabella alla ricerca di un'entry che contenga il numero di pagina virtuale che stiamo cercando di mappare associata al processo corrente.

   Questa tecnica è necessaria quando si ha a che fare con spazi di indirizzamento molto grandi (tipicamente a 64bit) e una page table ordinaria non potrebbe di conseguenza risiedere in memoria principale a causa del numero di entry troppo elevato.

3. Si consideri un sistema con memoria paginata a due livelli, le cui page table siano mantenute in memoria principale. Il tempo di accesso alla memoria principale sia $t = 30ns$.

   1. Qual è il tempo effettivo di accesso alla memoria?
   2. Aggiungendo un TLB, con tempo di accesso $\epsilon = 1ns$, quale hit rate dobbiamo avere per un degrado delle prestazioni del 10% rispetto a t?

   Risposta:

   1. Il tempo effettivo di accesso alla memoria è $3t$, ovvero, 90 ns; infatti sono necessari 30 ns per accedere alla page table esterna, 30 ns per accedere alla page table interna ed infine 30 ns per accedere alla locazione nel frame fisico in memoria.
   2. Un degrado del 10% rispetto a t significa un EAT pari a $1,1 \cdot t$, ovvero, 33 ns. Quindi si ha quanto segue (α rappresenta l'hit rate):

      $$\begin{array}{rcc} EAT & = & \epsilon + \alpha t + (1-\alpha)3t \\ 33 & = & 1 + 30\alpha + (1-\alpha) \cdot 90 \\ 33 & = & 91 - 60\alpha \end{array}$$

      da cui si ricava $\alpha = \frac{58}{60} = 0,97$ (97%).

4. Cosa si intende per Memory Mapped I/O (I/O mappato in memoria)? E per Memory Mapped File (file mappato in memoria)?

   Risposta: Con l'espressione Memory Mapped I/O si intende una tecnica per far comunicare il sistema operativo con i controller dei dispositivi tramite la riserva di una parte dello spazio degli indirizzi di memoria. In questo modo l'accesso a tali indirizzi non verrà considerato come un normale accesso a locazioni della memoria principale, ma servirà ad accedere direttamente ai registri ed ai buffer dei controller dei dispositivi. Il vantaggio di tale approccio consiste nel fatto che non è necessario disporre di istruzioni speciali di I/O, in quanto tali operazioni possono essere portate a termine tramite le solite operazioni di manipolazione della memoria applicate alla porzione dello spazio di indirizzi mappato.

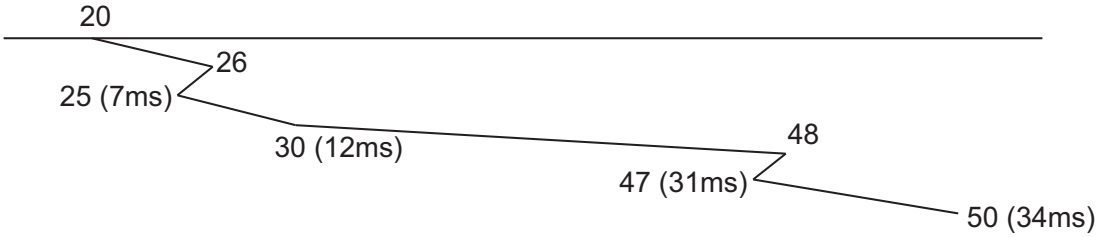
   Per Memory Mapped File invece si intende una tecnica per facilitare l'accesso ai file. Praticamente una zona della memoria viene riservata attraverso una speciale system call (`mmap`) in modo da ospitare il contenuto di un file. A questo punto, tramite le solite funzioni per la manipolazione della memoria, è possibile accedere e modificare il file senza dover accedere alla memoria secondaria. Alla fine tutte le modifiche verranno sincronizzate su disco, dato che il file viene visto come un'area di swap per la porzione di memoria allocata.

5. Si consideri un disco gestito con politica SSTF. Inizialmente la testina è posizionata sul cilindro 20; lo spostamento ad una traccia adiacente richiede 1 ms. Al driver di tale disco arrivano richieste per i cilindri 30, 25, 50, 47, rispettivamente agli istanti 0 ms, 6 ms, 9 ms, 30 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?

   Risposta:

   1. Le richieste vengono servite nell'ordine 25, 30, 47, 50, come illustrato dal seguente diagramma:



   2. Il tempo di attesa medio è dato da $\frac{(7-6)+(12-0)+(31-30)+(34-9)}{4} = \frac{1+12+1+25}{4} = \frac{39}{4} = 9,75ms.$

6. Spiegare brevemente come funziona una chiamata RPC.

   Risposta: Una Remote Procedure Call (RPC) consente ad un processo in esecuzione su un host di effettuare una chiamata ad una procedura che viene eseguita su un host remoto come se questa fosse una normale chiamata ad una funzione locale. Lo scambio di messaggi necessario per il funzionamento di una RPC è completamente invisibile al programmatore. Sostanzialmente quando un processo su un host A chiama una procedura su un host B, il processo chiamante su A viene sospeso, l'informazione necessaria per la computazione (i parametri della procedura) vengono comunicati via rete e l'esecuzione prosegue su B. Per effettuare una chiamata RPC il processo chiamante (client) utilizza una piccola procedura (client stub) che rappresenta la procedura remota nello spazio di indirizzamento del client. Il client stub organizza i parametri in un messaggio (marshaling) che viene spedito all'host remoto. Giunto a destinazione il messaggio, esso viene elaborato dal server stub che chiama la procedura responsabile della computazione. Il risultato viene poi rispedito via rete dal server stub al client stub che lo restituisce al processo chiamante. Quest'ultimo può quindi riprendere la sua esecuzione come in seguito ad una normale chiamata di procedura locale.

7. Descrivere brevemente le differenze principali fra multiprocessore, multicomputer e sistema distribuito.

Risposta: I sistemi multiprocessore sono sistemi strettamente accoppiati, ovvero, sono caratterizzati dalla condivisione di clock e/o memoria (es.: sistemi UMA, NUMA). I multicomputer sono sistemi strettamente accoppiati, che non condividono clock e/o memoria (ogni nodo è costituito da una propria CPU, memoria, scheda di rete ed a volte anche da un disco), ma che sono caratterizzati dalla presenza di linee di comunicazione molto veloci fra i vari nodi del sistema che solitamente risiedono nello stesso rack o stanza e che sono amministrati da una singola organizzazione. I sistemi distribuiti sono sistemi debolmente accoppiati, i cui nodi sono sistemi completi sparsi in regioni geograficamente anche molto distanti fra loro, controllati da diverse organizzazioni e connessi da reti con linee di comunicazione molto più lente (es.: Internet).

Sistemi Operativi
2 aprile 2007
Compitino 1/B

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Si descriva l'algoritmo di scheduling del sistema operativo Unix tradizionale.

   Risposta: L'algoritmo di scheduling del sistema operativo Unix tradizionale è di tipo RR con code multiple; ogni processo ha una priorità di scheduling e numeri più grandi indicano priorità minore. Per favorire i processi interattivi è previsto un feedback negativo sul tempo di CPU impiegato ed un meccanismo di invecchiamento per prevenire la starvation.

   Un processo esegue per il quanto di tempo assegnato; alla fine di quest'ultimo viene prelazionato. Quando il processo j rilascia la CPU:

   - viene incrementato il suo contatore CPU_j di uso CPU,
   - viene messo in fondo alla stessa coda di priorità,
   - riparte lo scheduler su tutte le code.

   Una volta al secondo, vengono ricalcolate tutte le priorità dei processi in user mode (dove $nice_j$ è un parametro fornito dall'utente):
   $CPU_j = CPU_j/2$ (*fading esponenziale*)
   $P_j = CPU_j + nice_j$

   I processi in kernel mode non cambiano priorità.

2. (a) Che cosa si intende nei sistemi operativi per "prelazione di un processo"? Un processo è sempre prelazionabile?

   (b) Si elenchino le situazioni in cui un processo viene prelazionato.

   Risposta:

   (a) Per "prelazione di un processo" si intende il fatto che lo scheduler lascia eseguire un processo per un certo periodo di tempo, passato il quale il processo viene sospeso in modo da permettere eventualmente ad un altro processo di usufruire del tempo di CPU. Per essere implementato presuppone il verificarsi di un clock interrupt allo scadere dell'intervallo di tempo concesso (altrimenti lo scheduler non può riprendere il controllo della CPU).

   Un processo non è sempre prelazionabile (basti pensare al caso di un processo real time con priorità maggiore di tutti gli altri processi real time).

   (b) Le situazioni in cui un processo viene prelazionato sono:

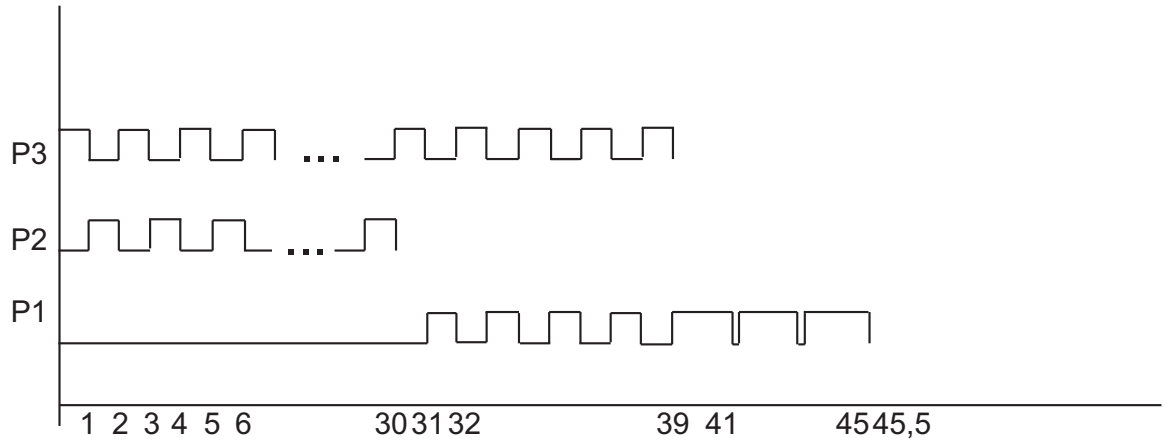
   - quando scade il quanto di tempo concesso per l'utilizzo della CPU;
   - quando arriva un nuovo processo nella coda ready (lo scheduler può mandare in esecuzione quest'ultimo al posto del processo prelazionato).

3. I processi P_1, P_2, P_3, di priorità crescente e con richieste di CPU totali di 10, 15, 20 s, sono in coda ready nell'ordine indicato. L'algoritmo di scheduling della CPU è con prelazione (e i processi prelazionati vengono messi in fondo alla coda ready). Ogni processo P_i ha un comportamento ciclico: in ogni ciclo consuma c_i secondi di CPU ed esegue operazioni di I/O per o_i secondi, dove $c_i = 2^{3-i}$ e $o_i = 0,5^{3-i}$. L'algoritmo di scheduling viene eseguito ad intervalli di 1 secondo. Si calcolino i tempi di attesa per i processi, nel caso l'algoritmo di scheduling sia RR con quanto pari a 2s e con prelazione in base alla priorità.

   Risposta: I tempi di attesa Δ_i dei processi P_i sono $\Delta_1 = 35,5s$, $\Delta_2 = 15s$ e $\Delta_3 = 19s$. Infatti abbiamo la seguente situazione:

	CPU$_t ot$	c_i	o_i
P_1	10	4	0,25
P_2	15	2	0,5
P_3	20	1	1

Sistemi Operativi
2 aprile 2007
Compitino 1/B



4. Si consideri la seguente situazione, dove P_0, P_1, P_2 sono tre processi in esecuzione, C è la matrice delle risorse correntemente allocate, Max è la matrice del numero massimo di risorse assegnabili ad ogni processo e A è il vettore delle risorse disponibili:

	C			Max		
	A	B	C	A	B	C
P_0	1	0	4	4	2	7
P_1	2	0	1	4	4	5
P_2	0	3	2	2	4	2

Available (A)		
A	B	C
2	1	1

   (a) Calcolare la matrice R delle richieste.

   (b) Il sistema si trova in uno stato sicuro?

   Risposta:

   (a) La matrice R delle richieste è data dalla differenza $Max - C$:

R		
A	B	C
3	2	3
2	4	4
2	1	0

   (b) No il sistema non è in uno stato sicuro in quanto non esiste una sequenza di esecuzione sicura. Infatti, dapprima si può eseguire soltanto P_2 in quanto R_2 è l'unica riga di R minore od uguale al vettore A che quindi assume il valore $(2, 4, 3)$. A questo punto non esiste nessuna riga rimanente di R minore od uguale al nuovo valore di A. Quindi il sistema è in stallo.

5. Si illustri la differenza fra elusione dei deadlock (deadlock avoidance) e prevenzione dei deadlock (deadlock prevention).

   Risposta: Per deadlock avoidance si intende una strategia di allocazione delle risorse che eviti l'insorgere di stalli, scegliendo oculatamente l'assegnamento delle risorse disponibili. Ad esempio l'algoritmo del banchiere è un algoritmo di elusione dei deadlock visto che consente l'allocazione di una risorsa soltanto se lo stato conseguente rimane uno stato sicuro.

   Per deadlock prevention invece si intende negare la possibilità dell'insorgere di stalli a priori, negando (almeno) una delle condizioni di Coffman (visto che sono condizioni necessarie per l'insorgere di un deadlock).

6. L'utilizzo dei semafori è sufficiente per escludere il problema del deadlock? In caso di risposta positiva si spieghi il motivo, mentre in caso di risposta negativa si fornisca un controesempio.

   Risposta: No, il solo utilizzo dei semafori non pone al riparo da possibili situazioni di stallo, come viene illustrato dai seguenti frammenti di codice di due ipotetici processi in esecuzione:

```
down(&resource_1);                    down(&resource_2);
down(&resource_2);                    down(&resource_1);
use_both_resources( );                use_both_resources( );
up(&resource_2);                      up(&resource_1);
up(&resource_1);                      up(&resource_2);
```

Nel caso in cui il primo processo riesca a bloccare `resource_1` ed il secondo processo riesca a bloccare `resource_2`, allora si verificherà un deadlock al momento di eseguire le `down` successive.

7. Si illustri il concetto di variabile condizione (condition variable) ed il funzionamento delle operazioni wait e signal nei monitor.

   Risposta: Le variabili condizione servono a gestire l'attesa e la segnalazione di eventi tramite le operazioni wait e signal. Più precisamente, un processo che voglia sospendersi in attesa del verificarsi di un evento legato ad una particolare condition variable effettuerà una wait su quest'ultima. La sospensione del processo permette ad un altro processo di entrare nel monitor. La signal invece viene utilizzata per segnalare il verificarsi di un evento associato ad una condition variable (utilizzata come argomento). La signal non fa altro che risvegliare uno dei processi in attesa di quella particolare variabile condizione. A questo punto, per evitare di avere due processi all'interno del monitor, ci sono due possibilità:

   - soluzione di Hoare: il processo che esegue la signal viene sospeso per permettere al processo risvegliato di ottenere il blocco sul monitor;
   - soluzione di Brinch Hansen: il processo che esegue la signal deve uscire immediatamente dal monitor, ovvero, una signal può occorrere soltanto come ultimo comando all'interno di una procedura.

Sistemi Operativi
2 aprile 2007
Compitino 2/B

Si risponda ai seguenti quesiti, giustificando le risposte.

1. Si illustri cosa si intende per allocazione contigua della memoria con partizionamento dinamico. Questa tecnica può soffrire di frammentazione interna? E di frammentazione esterna? Motivare le risposte.

   Risposta: Nel caso di allocazione contigua con partizionamento dinamico della memoria, quest'ultima viene suddivisa a tempo di esecuzione, allocando zone della dimensione esatta a contenere l'immagine del processo (quindi non ci può essere frammentazione interna). Il sistema operativo tiene traccia delle porzioni allocate e di quelle libere (dette "buchi"). Quando viene lanciato in esecuzione un nuovo processo gli viene allocata una zona di memoria in un buco sufficientemente grande a contenerlo. In questo caso quindi, a lungo andare, vi possono essere dei problemi di frammentazione esterna, ovvero, la memoria complessivamente libera può essere sufficiente a contenere l'immagine di un nuovo processo, ma, essendo frammentata in piccoli buchi, risulta inutilizzabile. Per ovviare a questo problema è possibile utilizzare una tecnica di compattazione della memoria, ovvero, si riordinano le zone di memoria allocate in modo da agglomerare tutti i buchi in un unico buco di dimensioni sufficienti a soddisfare le richieste di un nuovo processo.

2. Cosa sono i TLB (Translation Lookaside Buffer)? Che utilità hanno?

   Risposta: I TLB sono una memoria associativa molto costosa e quindi presente in piccole quantità; serve a contenere le entry della page table relative alle pagine accedute più recentemente. Essendo in grado di compiere un confronto simultaneo su tutto il suo contenuto (in parallelo), permette di abbattere in modo significativo i tempi di traduzione da indirizzo logico ad indirizzo fisico.

3. Si consideri un sistema con memoria paginata ad un livello, la cui page table sia mantenuta in memoria principale. Il tempo di accesso alla memoria principale sia $t = 40ns$.

   1. Qual è il tempo effettivo di accesso alla memoria?
   2. Aggiungendo un TLB, con tempo di accesso $\epsilon = 1ns$, quale hit rate dobbiamo avere per un degrado delle prestazioni del 20% rispetto a t?

   Risposta:

   1. Il tempo effettivo di accesso alla memoria è $2t$, ovvero, 80 ns; infatti sono necessari 40 ns per accedere alla page table e 40 ns per accedere alla locazione nel frame fisico in memoria.
   2. Un degrado del 20% rispetto a t significa un EAT pari a $1,2 \cdot t$, ovvero, 48 ns. Quindi si ha quanto segue (α rappresenta l'hit rate):

      $$\begin{array}{rcc} EAT & = & \epsilon + \alpha t + (1-\alpha)2t \\ 48 & = & 1 + 40\alpha + (1-\alpha) \cdot 80 \\ 48 & = & 81 - 40\alpha \end{array}$$

      da cui si ricava $\alpha = \frac{33}{40} = 0,83$ (83%).

4. Illustrare brevemente le differenze fra:

   1. Programmed I/O,
   2. I/O guidato dagli interrupt senza DMA,
   3. I/O guidato dagli interrupt con DMA.

   Risposta: Mentre con la modalità Programmed I/O (I/O a interrogazione ciclica) il processore manda un comando di I/O e poi attende che l'operazione sia terminata, testando lo stato del dispositivo con un loop busy-wait (polling), con la modalità Interrupt-driven I/O, una volta inviato il comando di I/O, il processo viene sospeso fintanto che non arriva un interrupt a segnalare il completamento dell'operazione. Durante la sospensione del processo, la CPU può mandare in esecuzione altri processi o thread. Di fondamentale importanza è il vettore di interrupt che consente di selezionare la routine di gestione opportuna per ogni tipo di interrupt. Ovviamente la prima modalità è efficiente soltando nel caso in cui la velocità del dispositivo di I/O sia paragonabile a quella della CPU. La modalità DMA richiede un controller DMA e funziona in questo modo: la CPU imposta i registri del controller DMA specificando il tipo di azione di I/O, l'indirizzo di memoria ed il conteggio di

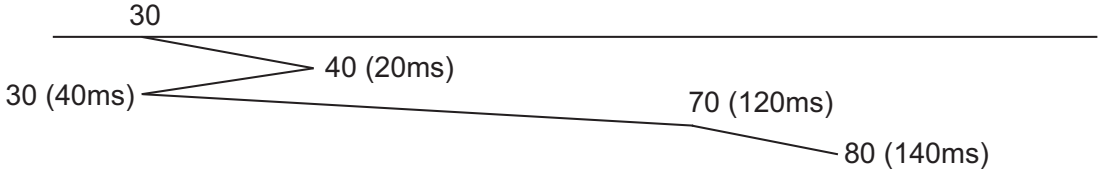
byte da trasferire. Poi i dati vengono trasferiti senza più richiedere l'intervento della CPU; infatti il controller del dispositivo di I/O riceve le richieste di lettura o scrittura da parte del controller DMA a cui notifica il completamento dell'operazione una volta che ha trasferito il byte da/verso l'indirizzo di memoria corretto (specificato dal controller DMA). A questo punto il controller DMA incrementa l'indirizzo di memoria comunicandolo sul bus e decrementa il conteggio dei byte da traferire, ripetendo la richiesta di lettura o scrittura al controller del dispositivo fintanto che il conteggio dei byte non raggiungerà lo zero. Soltanto a questo punto verrà inviato un interrupt alla CPU che potrà far ripartire il processo sospeso. Siccome il controller DMA deve bloccare il bus per consentire i trasferimenti dal controller del dispositivo alla memoria, se anche la CPU ha bisogno di accedere al bus dovrà aspettare, venendo così rallentata.

5. Si consideri un disco gestito con politica SSTF. Inizialmente la testina è posizionata sul cilindro 30, ascendente; lo spostamento ad una traccia adiacente richiede 2 ms. Al driver di tale disco arrivano richieste per i cilindri 70, 40, 30, 80, rispettivamente agli istanti 0 ms, 10 ms, 18 ms, 60 ms. Si trascuri il tempo di latenza.

   1. In quale ordine vengono servite le richieste?
   2. Il tempo di attesa di una richiesta è il tempo che intercorre dal momento in cui è sottoposta al driver a quando viene effettivamente servita. Qual è il tempo di attesa medio per le quattro richieste in oggetto?

   Risposta:

   1. Le richieste vengono servite nell'ordine 40, 30, 70, 80, come illustrato dal seguente diagramma:



   2. Il tempo di attesa medio è dato da $\frac{(20-10)+(40-18)+(120-0)+(140-60)}{4} = \frac{10+22+120+80}{4} = \frac{232}{4} = 58ms$.

6. Illustrare la differenza fondamentale fra sistemi strettamente accoppiati e debolmente accoppiati, elencando degli esempi per ognuna delle due categorie.

   Risposta: I sistemi strettamente accoppiati sono caratterizzati dalla condivisione di clock e/o memoria (es.: sistemi UMA, NUMA) oppure dalla presenza di linee di comunicazione molto veloci fra i vari nodi del sistema che solitamente risiedono nello stesso rack o stanza e che sono amministrati da una singola organizzazione (es.: multicomputer, cluster of workstations). Viceversa i sistemi debolmente accoppiati sono costituiti da sistemi completi (in cui ogni nodo è sostanzialmente una macchina completa) sparsi in regioni geograficamente anche molto distanti fra loro, controllati da diverse organizzazioni e connessi da reti con linee di comunicazione molto più lente (es.: Internet).

7. Descrivere brevemente cos'è una socket ed elencarne almeno due tipi diversi.

   Risposta: Una socket rappresenta un *endpoint* di comunicazione e solitamente è associata ad un indirizzo la cui natura dipende dal dominio di comunicazione scelto (processi che comunicano nello stesso dominio utilizzano lo stesso formato di indirizzi). I domini più comuni sono il dominio Unix (`AF_UNIX`), il dominio Internet (`AF_INET`, `AF_INET6`) il dominio Novell (`AF_IPX`) e il dominio AppleTalk (`AF_APPLETALK`).

   Esistono diversi tipi di socket che rappresentano diverse classi di servizi (non tutti sono disponibili in ogni dominio di comunicazione):

   - stream socket: forniscono stream di dati affidabili, duplex, ordinati,
   - socket per pacchetti in sequenza: forniscono stream di dati, ma i confini dei pacchetti sono preservati,
   - socket a datagrammi: trasferiscono messaggi di dimensione variabile, preservando i confini, ma senza garantire ordine o arrivo dei pacchetti,

- socket per datagrammi affidabili: come quelle a datagrammi, ma l'arrivo è garantito,
- socket raw: permettono di accedere direttamente ai protocolli che supportano gli altri tipi di socket: sono utili per sviluppare nuovi protocolli.